APRENENTATGE COMPUTACIONAL

PRÀCTICA 1

AUTORS: JAN MOROS **1492333**
ADRIÀ CARRASQUILLA **1492104**

CURS: 3R ENGINYERIA INFORMÀTICA
MENCIÓ EN COMPUTACIÓ

DATA: 27/10/2019

ÍNDEX

1. INTRODUCCIÓ

El títol d'aquesta pràctica és Regressió-Classificació, i aquests seran els dos eixos principals del projecte.

Entenem la regressió com un model que serveix per fer prediccions sobre valors numèrics, a partir de la combinació de variables que defineixen el problema que s'intenta resoldre. Ens serveix, a més, per analitzar el comportament de les dades, és a dir, estudiar la relació entre les variables, emfatitzant la relació amb l'objectiu, la variable que s'intenta predir. I, a partir d'aquesta última, determinar quines són les variables més importants del problema.

Pel que fa a la classificació, és un model que a partir d'unes dades categoritzables en diverses classes, té com a sortida una funció classificadora que divideix l'espai en àrees per a cada classe. D'aquesta manera, si entrem noves dades a la funció, ens permet predir a quina classe estan assignades.

La pràctica es divideix en tres apartats, C, B i A. En aquesta entrega arribem fins a l'apartat B.

El primer apartat consisteix en analitzar la base de dades. Tenir molt clar com està estructurada i sobretot la relació que existeix entre els atributs ens serà molt útil per fer el seu tractament, ja que podem obviar variables que trobem que no aporten informació, o que no tenen cap tipus de relació amb la variable objectiu. També, depenent de la base de dades, és en aquest apartat on es decideix quin és, l'atribut objectiu. No és el nostre cas, perquè a la base de dades que se'ns ha assignat queda molt clar quina és la variable objectiu.
A part d'analitzar les dades, a l'apartat C també se'ns demana que fem una primera prova de regressió, utilitzant el regressor lineal de la llibreria sklearn, i n'analitzem els resultats.

A l'apartat B la tasca a realitzar és implementar un model més avançat de regressor, basat en el descens de gradient. Això ens permetrà fer regressors que s'adaptin millor al problema, que és altament probable que no segueixi un model lineal. Un cop dissenyat el model, que ens permetrà jugar amb els seus paràmetres per trobar la millor configuració, haurem d'avaluar-lo també, comparant-lo amb els resultats del regressor lineal de l'apartat C.

Tots aquests resultats, per poder-los analitzar més fàcilment i per mostrar-los a un suposat client, els representarem en taules i gràfiques. D'aquesta manera podem veure ràpidament aspectes com la relació d'un atribut amb un altre o la forma aproximada que segueixen les mostres.

2. RESOLUCIÓ DEL PROBLEMA

ANÀLISIS BASE DE DADES

Tot i que no hi ha un apartat específic en l'enunciat de la pràctica on s'indica que s'hagi de fer un estudi sobre el dataset, considerem que és un pas importantíssim no tant per a la resolució del problema però si per a la interpretació dels seus resultats.

En el nostre cas, la base de dades a estudiar s'anomena **Bike Sharing Dataset**. Conté les dades recollides en dos anys sobre el lloguer de bicicletes d'una empresa situada a Washington DC. Aquests atributs són els següents:

- **instant:** Identificador mostra
- **dteday:** Data de la mostra
- **season:** Estació (1: hivern, 2: primavera, 3: estiu, 4: tardor)
- **yr:** any (0: 2011, 1:2012)
- **mnth:** Mes (1 a 12)
- **hr:** Hora (0 a 23)
- **holiday:** Si és festiu o no
- **weekday:** Dia de la setmana
- **workingday:** Si és dia laboral o no
- **weathersit:** temporal (1: clar o mitjanament ennuvolat, 2: boira i núvols, 3: precipitacions lleugeres, 4: precipitacions elevades)
- **temp:** Temperatura normalitzada en graus Celsius.
- **atemp:** Sensació de temperatura normalitzada en graus Celsius.
- **hum:** Valor humitat normalitzat
- **windspeed:** Velocitat del vent normalitzada
- **casual:** Recompte del nombre de bicicletes llogades per una persona casual
- **registered:** Recompte del nombre de bicicletes llogades per un usuari registrat
- **cnt:** Recompte total de les bicicletes llogades

Aleshores, un cop observats els atributs que acabem de explicar, ja es pot decidir quin seria l'atribut objectiu a predir utilitzant el regressor. En el nostre cas hem considerat que el més indicat és **cnt**, doncs el que ens sembla més lògic és que a

partir d'una sèrie de dades sobre un dia i hora (temperatura, tipus de dia, si es festiu...), predir quantes bicicletes es llogaran. D'aquesta manera en cas de que fóssim l'empresa proveïdora d'aquests vehicles, podríem saber quan necessitarem més bicicletes o si n'estem oferint massa en relació al que realment s'utilitza i així optimitzar el cost i qualitat del servei.

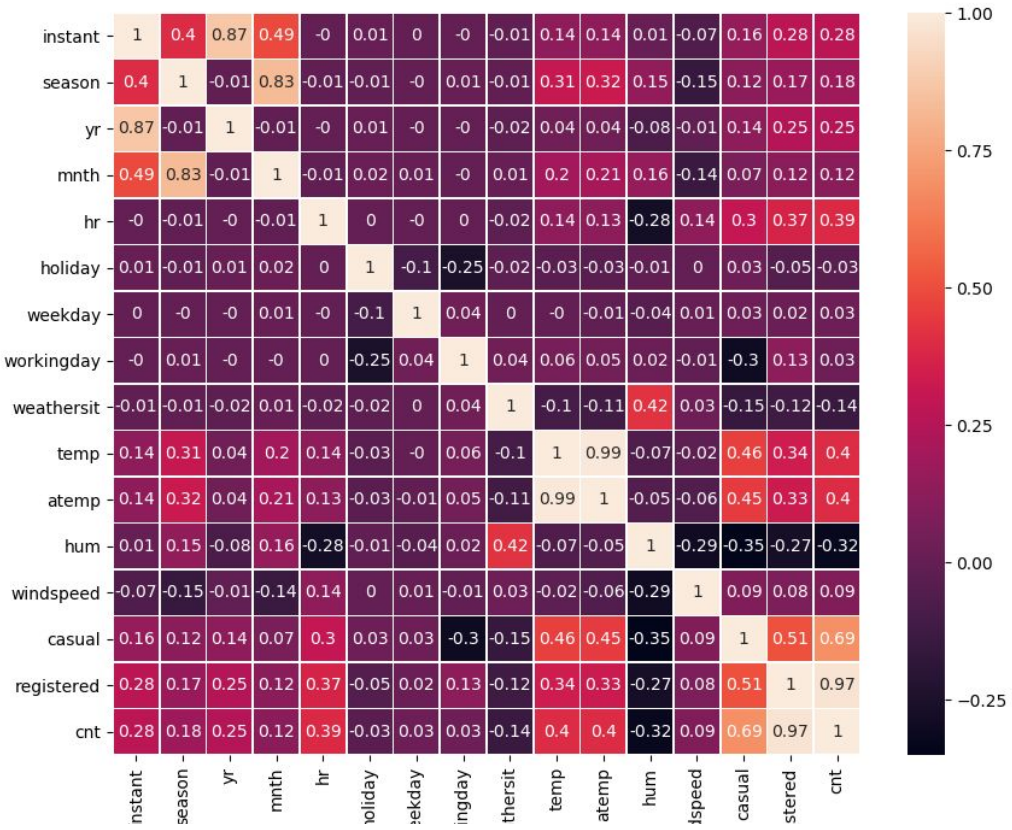
Tanmateix, només amb la descripció donada dels atributs, també en podem descartar alguns de redundants.

En el cas de **registered** i **casual** la informació que rebem no deixa de ser un desglossament de l'atribut **cnt.** Per aquesta raó considerem que podem prescindir d'aquestes dues columnes ja que ens interessa reduir el nombre d'atributs amb la finalitat de tenir un model de major rendiment.

Per altra banda **atemp** i **temp** ens donen pràcticament la mateixa informació ja que entre la temperatura real i la sensació de temperatura la diferència sol ser mínima i amb una correlació molt elevada. Per tant, podem prescindir d'un dels dos, en el nostre cas hem optat per mantenir **temp.**

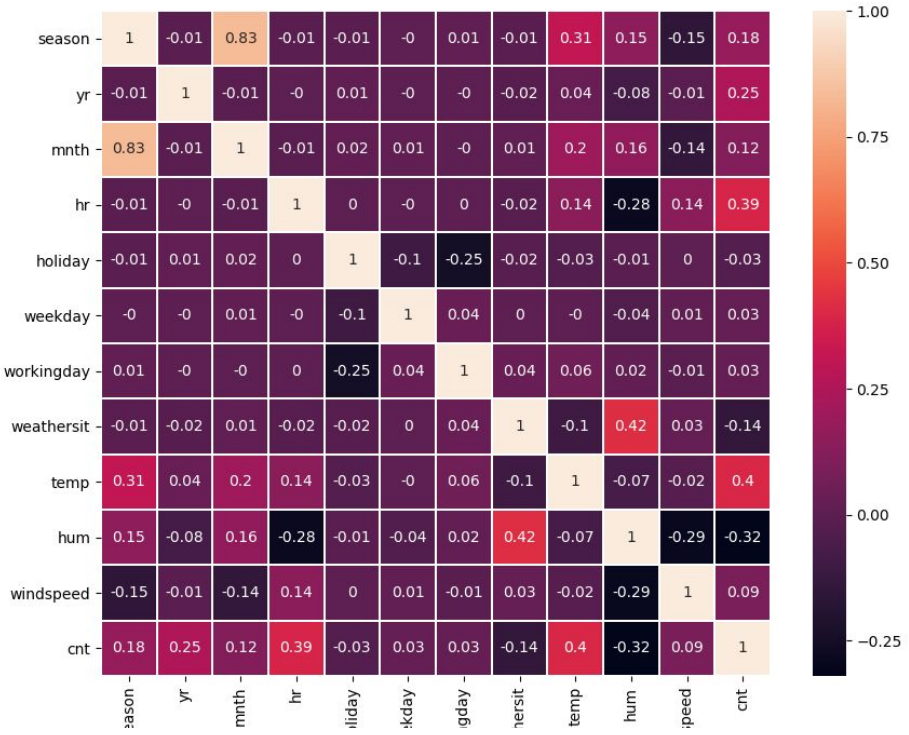
L'atribut **instant** també és completament innecesari, doncs l'identificador de la mostra no ens aportarà cap informació rellevant (és un enter que s'incrementa en u per mostra). Simultàniament podem suprimir **dteday** ja que a part de que ens ve donat en format string i el seu tractament no pot efectuar-se de manera directa, la informació que ens aporta es troba desglossada en altres atributs com **season, mnth, weekday, yr**.

Per reforçar aquesta neteja d'atributs, vam utilitzar un heatmap per a poder estudiar la correlació que hi ha entre variables.



Com podem observar, les relacions entre atributs ja comentades es justifiquen amb la correlació. Els casos més evidents són el de **atemp** i **temp** i el de **registered** i **cnt.**

Un cop eliminem aquells atributs que hem considerat irrellevants, el heatmap passa a quedar de la següent manera:



Amb aquest nou heatmap reduït podem analitzar amb més claredat altres característiques que se'ns poden haver escapat.

El primer tret a destacar és la alta correlació entre **mnth** i **season**. És una relació prou evident, doncs les estacions estan delimitades pels mesos de l'any. Els següents amb un valor també elevat són **hum** i **weathersit**. De nou, és un resultat que no ens sorprèn. És d'allò més normal que la humitat vagi lligada a si hi ha precipitacions, boira o és un dia clar.

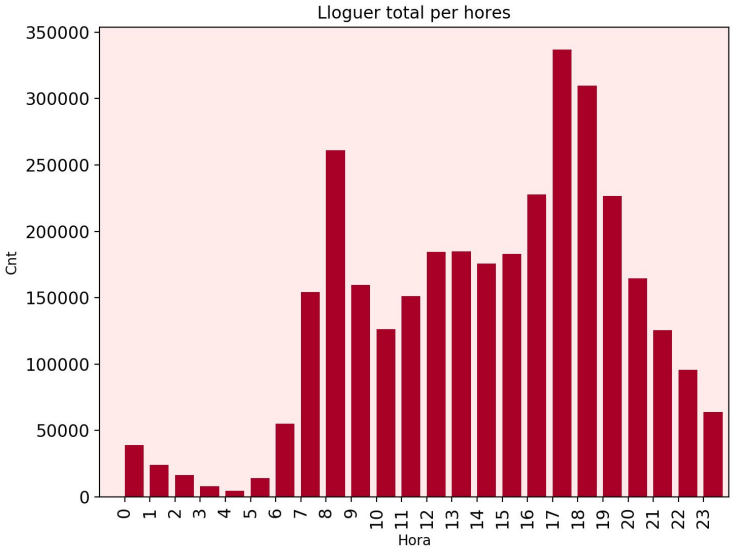
Però amb aquest heatmap ens interessa bastant veure quins són els atributs que més relació tenen amb el nostre target, probablement seran aquells als que haurem de tractar amb més profunditat. Podem destacar:

- **season:** Com a hipòtesis podem pensar que la estació de l'any influeix en la quantitat de bicicletes llogades perquè cada estació engloba una sèrie de condicions que afavoreixen o no aquest servei. Els hiverns i tardors són freds i amb més precipitacions que no pas l'estiu o la primavera i potser això influeix en els usuaris de Washington DC.
- **yr:** Podem deduir que per raons de maduració del negoci, hi ha un augment total en l'ús d'aquest servei d'un any a l'altre.
- **hr:** També resulta evident que l'hora sigui conjuntament amb temp el factor amb més relació, doncs hi ha hores puntes on la gent s'ha de

desplaçar. És més normal que tinguem sempre un nombre més elevat de lloguer a les 8 del matí que no pas a les 3 de la matinada.

- **temp:** Aquest és l'atribut amb el valor de correlació més elevat. Podem pensar que a més calor, més agradable és agafar una bicicleta. També podem pensar que quan tenim precipitations, les temperatures també disminueixen i quan el dia és més aviat assolejat, aquest valor tendeix a augmentar.

Anem a aprofundir una mica més en aquests atributs per deixar de banda les especulacions i poder visualitzar els valors.



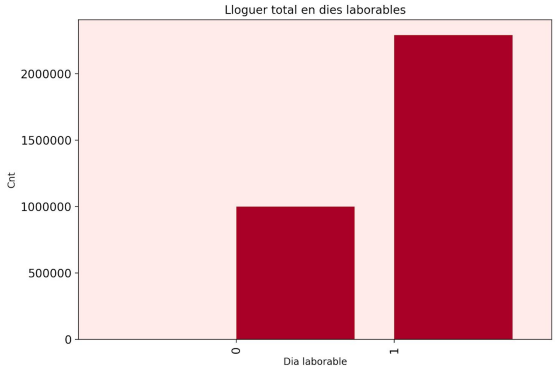
En aquest primer gràfic hem plasmat el número de bicis en total que s'han llogat en tot el transcurs de l'estudi que reflecteix la base de dades dividint-ho en les diferents hores del dia.

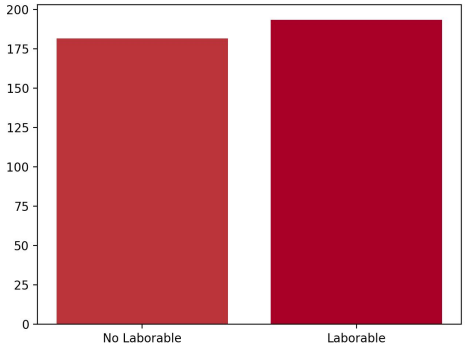
La distribució és força esperable. Tenim una quantitat molt *baixa* de lloguers en *horari nocturn* (de 22 a 6) i uns valors més *elevats* per a les hores de la *jornada habitual* de qualsevol persona. A més a més es pot apreciar clarament la presencia d'hores puntes. Podríem dividir-ho en dues franges:

- **De 7 a 9 del matí**
- **De 5 a 8 de la tarda**

Amb aquestes dades ja podem anar perfilant una mica més per on pot anar el comportament del tipus de persona que usa aquest servei. Aquestes dues

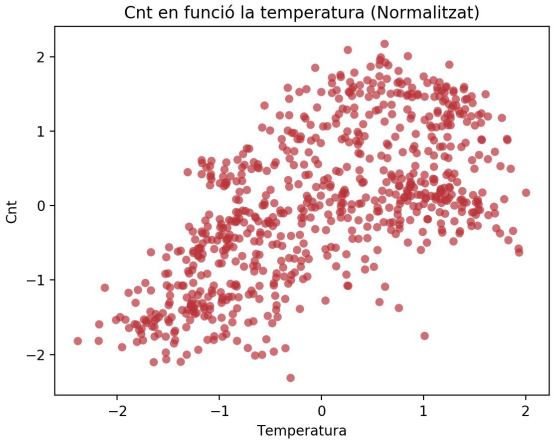
franges coincideixen amb el que en la majoria de casos són l'hora de *posar-se en marxa* i l'hora de *tornar a casa*. Entenem posar-se en marxa com a anar a treballar, estudiar, o a realitzar activitats habituals en un horari comú. Per tant això també ens dóna a entendre que potser és un servei que s'utilitza més habitualment al dia a dia (jornades laborals) i potser no tant en festius. Anem a comprovar aquesta hipòtesi:

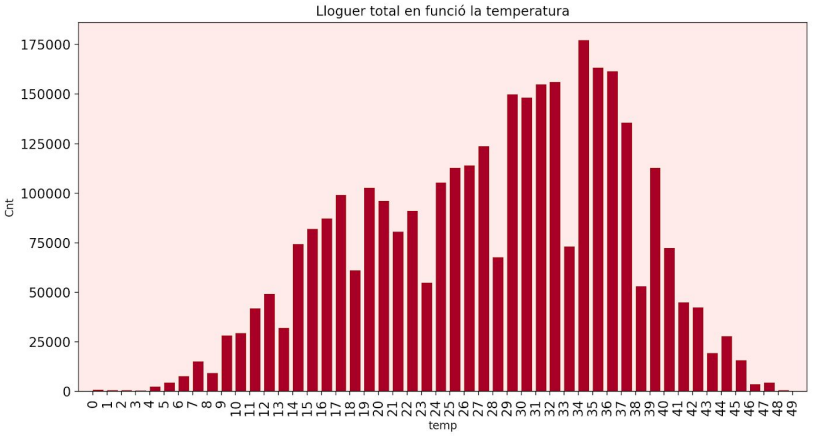




Pel que sembla, aquesta hipòtesi ha estat un pèl mal encaminada. Si només tenim en compte el gràfic de l'esquerra sembla que és el que pensàvem inicialment: s'han llogat moltes més bicicletes en dies laborables que no pas en els que no ho son. Tot i això, aquesta ha estat l'estratègia de representació seguida en la gràfica del lloguer per hores, però ara no ens dona la mateixa validesa d'informació. Cada dia té el mateix nombre d'hores pero un any no té el mateix nombre de dies laborables que de no-laborables. Per això hem afegit el segon gràfic on tenim la mitja de bicicletes llogades en cada tipus de dia. La diferència és prou petita i per tant podem deduir que *és un servei usat de manera força uniforme al llarg de l'any* tot i que té més pes en els dies laborals (sobretot pel fet que és una porció de l'any majoritària).

A continuació ens disposem a analitzar els valors de la temperatura.

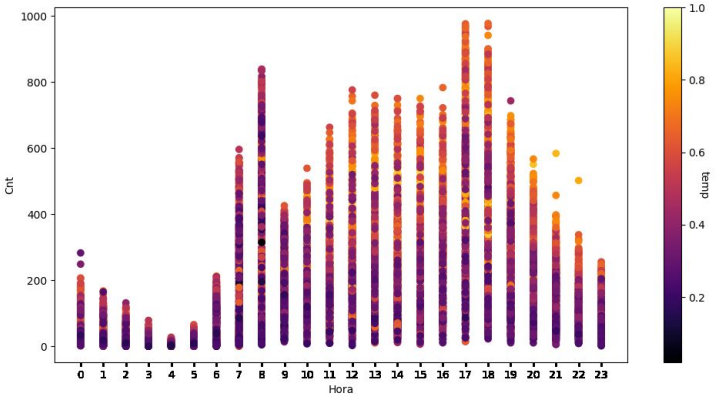




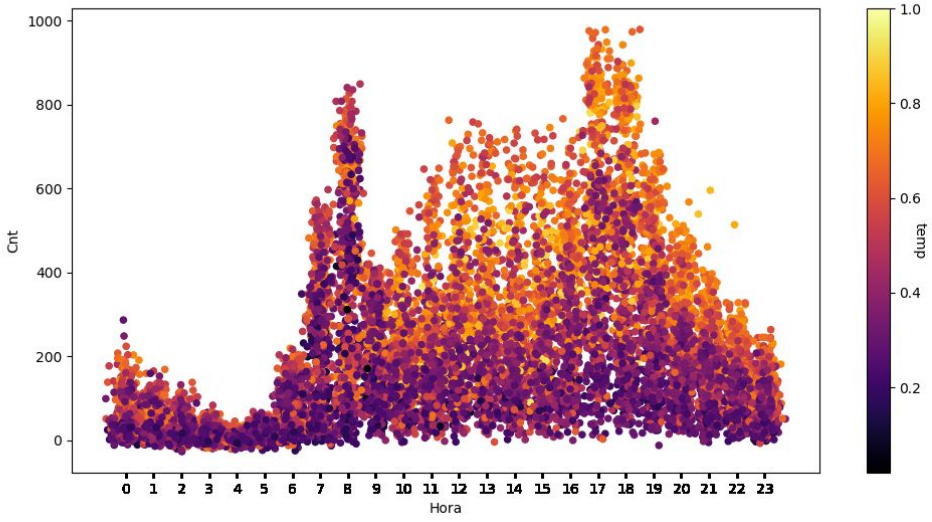
Amb aquestes dues gràfiques podem veure representada la relació entre la **temperatura** i el **cnt** total. En el primer cas (*esquerre*) tenim un scatter plot que ens mostra una relació més directa. El segon gràfic plasma el total de bicicletes llogades en una temperatura determinada. S'observa com per a temperatures molt baixes i molt altes l'ús del servei és bastant reduït. D'aquesta observació, però, cal tenir en compte que el nombre de mostres segueix una distribució propera a la normal: els valors més repetits es troben al centre (entre 20 i 40) mentre que els menys repetits són a les cues de la campana gaussiana. En certa part això explica la forma del segon gràfic.

Tot i això, tant en un com en l'altre podem intuir un comportament: **a temperatures més elevades, més lloguer de bicicletes**.

Per a reforçar aquesta conclusió farem una última representació. Aquest cop volem veure de nou la relació entre els atributs **cnt** i **hr** però a més li afegirem com a tercera variable **temp.**



Tot i que ens podem fer una idea amb aquest gràfic, la representació no és la més agradable a la vista. Això es deu a que el rang de **hr** és discret. Per a solucionar-ho, hem definit una funció jitter que altera la posició en x de manera que sembla ser una variable contínua. Així es veuen amb més claredat els resultats.



Com podem observar, hi ha una clara relació entre aquests tres atributs. Veiem com quan **cnt** *és menor sol tenir una temperatura menor* (sigui quina sigui la hora) i en canvi una *temperatura major pels seus valors més elevats*. També veiem com solen ser més fredes les hores amb menys quantitat de lloguer. L'excepció d'aquest cas és la primera franga d'hora punta, doncs a les 7-8 del matí les temperatures no solen ser gaire elevades. Amb això es reforça la nostra idea de que el servei té un gran pes en aquelles persones amb responsabilitats com treballar o estudiar. Per l'altra banda, a la franja de les 5-7 tenim els valors més alts de **cnt** alhora que les temperatures són també força elevades.

Fins aquí considerem que hem fet un estudi força exhaustiu de la base de dades i que tenim una idea prou encaminada sobre el comportament de les seves dades. Com a conclusió inicial podem considerar que probablement els atributs més rellevants per a fer la regressió i poder fer prediccions són els recentment comentats. Per tant, considerem finalitzat l'anàlisis inicial i ens endinsem amb la formulació dels diferents models d'aprenentatge.

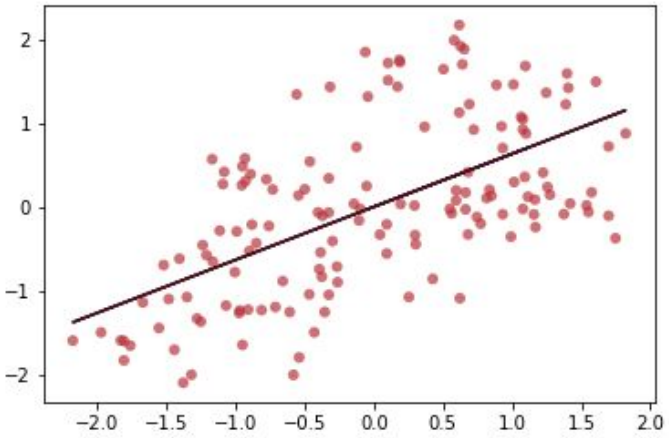
APARTAT C: REGRESSOR LINEAL

Un cop fet l'anàlisi exhaustiu de la base de dades, i tenint clar com s'estructura i com es relacionen els seus atributs entre ells, és el moment d'aplicar la regressió.
Per fer-ho, en aquest primer apartat, farem servir el regressor lineal de la llibreria de machine learning de python **scikit learn.** Fem servir aquest tipus de regressor tan simple com un primer contacte amb la base de dades, per veure si les dades tenen la forma apropiada per aplicar un regressor lineal.
Per tal de comprovar-ho, apliquem directament el regressor per veure quin resultat ens dóna.
Degut a que utilitzem 10 atributs, necessitaríem un espai de 10 dimensions per representar els resultats, cosa que és impossible en el nostre univers tridimensional. Per aquesta raó, no podem visualitzar les dades de manera gràfica, però sí que podem calcular el *coeficient de determinació*. Aquest coeficient determina la qualitat del model per predir resultats, i es defineix com la proporció de la variança en la variable dependent que pot ser explicada a partir de les independents. Aquest pren valors habitualment entre 0 i 1, sent un 1 que la predicció s'ajusta perfectament als valors reals. Hi ha alguns casos, però, en els que pot arribar a ser negatiu, i és en aquells que no segueixen la forma de les dades, més concretament quan prediuen pitjor que una funció constant.
D'aquesta manera, quan apliquem el regressor lineal ens dóna un coeficient del **0,7988.** Considerem que no és un valor dolent, s'ajusta de certa manera a les dades. Tot i això, no és el que seria d'esperar amb un regressor. El seu valor òptim oscil·la al voltant del **0,97**, que s'allunya bastant de l'obtingut.
Per tal d'estandaritzar l'ordre de magnitud de les variables, s'acostuma a seguir un procés de normalització de les dades. Avaluant el regressor de nou amb les dades normalitzades, veiem que l'error mitjà al quadrat **Mean Squared Error**, una altra mesura de rendiment pel regressor, disminueix considerablement, de **753878** a **0,20**, el coeficient R2 segueix sent el mateix, pel que realment amb les nostres dades no afecta normalitzar al funcionament del regressor. De tota manera, a partir d'ara tractarem les dades normalitzades, per si en futures implementacions acaba influint.

Per a assegurar-nos que el model s'ajusta bé a dades noves, diferents a les d'entrenament, es separen les dades en tres conjunts: entrenament, validació i test. Es fa l'entrenament sobre les dades d'entrenament, que es valida amb les de validació i s'ajusta segons els resultats. Llavors, seguint diferents criteris, es canvien els atributs del regressor i es torna a aplicar, agafant dades diferents per l'entrenament i la validació. Un cop s'està satisfet amb la configuració del model, s'aplica a les dades de tes, mai vistes a l'entrenament, per comprovar-ne la predicció. En el nostre cas, farem un model més senzill, únicament dividint les dades en entrenament i validació, en proporció de 80/20.
Un cop separades les dades, apliquem el regressor només tenint en compte un atribut cada cop, per a tots els atributs, per avaluar-ne la idoneïtat.

Atribut	**Mean Square Error**	**R2 score**
season	0.8873	0.1613
year	0.76	0.2815
month	1.0292	0.027
holiday	1.0781	-0.019
weekday	1.0544	0.0034
workingday	1.07	-0.0115
weathersit	0.9681	0.085
temp	0.5657	0.4653
hum	1.063	-0.0047
windspeed	1.0216	0.0344

Com podem observar, els atributs amb menys error i, per tant, millor R2 score son **temp i year.** Aquest resultat és coherent amb l'estudi de les dades, que ens deia que eren dels atributs amb més correlació amb l'objectiu. A l'annex es poden trobar totes les gràfiques de correlació, i mostrem aquí la que té millor coeficient, que és també de les poques vairables continues de la base de dades, pel que és més fàcil que pugui ser lineal, comparada amb la resta de discretes.



cnt en funció de la temperatura, normalitzats i amb la recta de regressió

A la taula podem observar també com hi ha moltes variables que es podrien considerar sense relació amb l'objectiu, algunes fins i tot tenen el coeficient negatiu. Seria lògic, doncs, eliminar-les del nostre model. Aquesta era la nostra hipòtesi, però vam comprovar que realment no era així, perquè ens donava molt pitjor l'R2 score general si els trèiem. Això es deu, hem concluït, a que no tenen una relació directa amb l'atribut objectiu, però que combinats amb altres atributs sí que sumen per a la predicció.

APARTAT B: DESCENS DEL GRADIENT

Per al segon apartat de la pràctica es proposa canviar la metodogia per a trobar un regresor lineal dels atributs a la nostra base de dades. Concretament es demana l'ús del descens del gradient.

Abans de començar amb la resolució d'aquest apartat volem comentar que per a aquesta versió de la memòria no ha pogut estar completat al 100%. Tot i això volem plasmar el que ha estat treballat fins aleshores, els motius pels quals estem en l'estat que estem i les seves respectives conclusions.

Tot i que a l'enunciat es proposa una estructura determinada per a l'aplicació del regresor utilitzant el descens del gradient, nosaltres hem optat per als primers estudis crear una funció únicament ja que els subpassos a dur a terme no són gaire llargs. Aquesta funció és la següent:

```
def Regressor(x, y, arrayTheta, max_iter, epsilon, aplha):
    i = 1
    millora=epsilon+1
    costAnt=0
    while( i < max_iter and millora > epsilon):
        predict = np.dot(x, arrayTheta)
        loss = predict - y
        cost = np.sum(loss ** 2) / (2 * x.shape[0])
        millora = abs(costAnt-cost)
        costAnt = cost
        gradient = np.dot(x.T, loss) / x.shape[0]
        print("Iter: "+str(i)+"  Cost: "+str(cost))
        arrayTheta = arrayTheta - alpha * gradient
        i+=1
    return arrayTheta
```

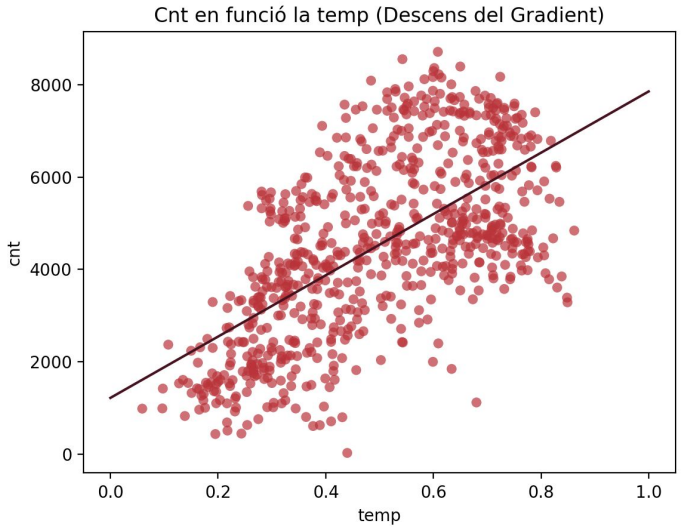
Li passem com a paràmetres els valors de l'atribut a analitzar **x**, els valors a predir **y**, un vector amb els valors de les thetes a fixar ja inicialitzades **arrayTheta**, el nombre màxim d'iteracions **max_iter**, l'error de convergencia **epsilon** i el coeficient d'aprenentatge **alpha**.

Mentre no es superin les iteracions màximes o la millora de cost sigui major a epsilon fem una predicció amb els valors actuals de theta, calculem l'error i el cost que genera i d'aquesta manera actualitzem els valors de theta. Un cop fora del bucle retornem els valors convergits en l'array de thetes.

Per a comprovar la seva correcta funció hem fet proves inicials amb poques dades inventades. Un cop satisfets del resultat, hem passat a provar-lo amb el nostre dataset.

Com ja hem destacat en l'anterior apartat, l'atribut amb millors característiques per a realitzar la regressió és la **temp**. Per tant, només ens cal usar la funció

implementada amb les dades d'aquest atribut en funció del target (**cnt**) i comprovar el resultat.



Els paràmetres utilitzats han estat:

alpha = 0.4 **max_it = 1000** **epsilon= 0.1**

El total d'iteracions necessàries per a la convergència han estat 529, la meitat de les màximes establertes. La recta que defineix la regressió y = ax + b té com a valors:

a = 6624.46 **b = 1222.91**

Després de fer vàries proves variant els paràmetres d'entrada hem comprovat que el cost mínim es troba sempre donat per una recta com la mostrada. El resulat obtingut és idèntic a l'obtingut en l'activitat anterior. Considerem vàlida la resolució.

Tot i això, aquest apartat ens ha donat més problemes dels esperats. Després de molta implementació errònia, tot i obtenir un model que aplicava el descens del gradient, els resultats no ens semblaven gens raonables, doncs la recta resultant s'allunyava molt de la zona on hi havia més concentració de valors. Al final ha resultat ser el problema més ximple que ens podia passar: vam considerar l'ordre equivocat d'utilitzar les thetes i per tant estavem representat la recta $y = bx + a$. No va ser fins que vam provar de utilitzar funcions de major grau que no ens vam adonar d'aquest error.

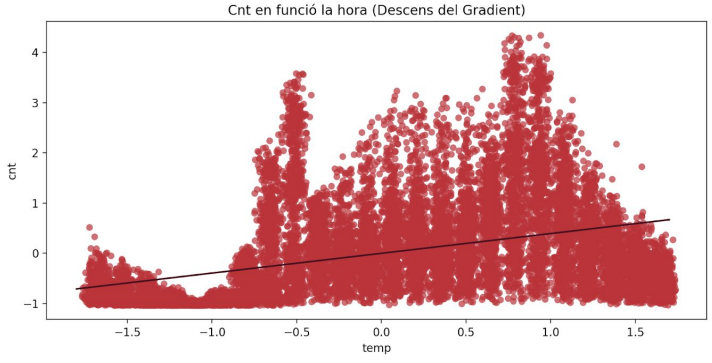
Un cop superada aquesta petita barrera ja ens podem dedicar a analitzar el comportament del model en funció dels paràmetres aportats. Aquest apartat no va poder ser completat en l'anterior entrega, però per aquesta ja l'hem enllestit.

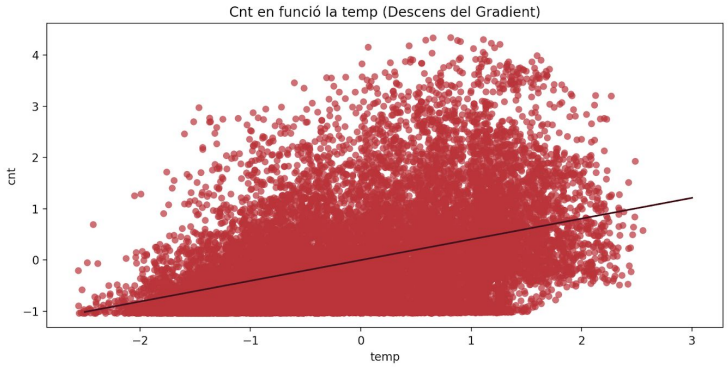
El primer que vam fer va ser passar a utilitzar el dataset **hour** on tenim cada instància separada per una hora determinada de cada dia. Vam començar a tenir algún petit problema amb la funció de *Regressor*, així que ens vam replantejar tornar a l'estructura aportada en els notebooks i omplir-la de manera funcional. La implementació és la següent:

```
class RegressorO(object):
    def __init__(self, arrayTheta, alpha, x, y, maxIt, epsilon, reg):
        # Inicialització segons els paràmetres
        ...
        #atributs per analitzar comportament dels paràmetres
        self.costos=[]
        self.iterations=[]

    def predict(self):
        loss = np.dot(self.x, self.arrayTheta)
        cost = np.sum(loss ** 2) / (2 * self.x.shape[0])
        return loss

    def __update(self, hy):
        self.arrayTheta = self.arrayTheta*(1-self.alpha*
(self.epsilon/self.x.shape[0])) -(1/self.y.shape[0])*alpha*( self.x.T.dot((hy -
self.y)))

    def train(self):
        i=0
        millora=epsilon+1
        costAnt=0
        while( (i <= self.maxIt) and (millora > self.epsilon)):
            self.it+=1
            pred = self.predict()
            loss=pred-self.y
            self.cost=(np.sum(loss**2)+reg*np.sum(thetas**2))/(2*self.x.shape[0])
            self.costos.append(float(self.cost)*100)
            self.iterations.append(self.it)
            millora = abs(self.cost-costAnt)
            self.__update(pred)
            costAnt=self.cost
            print("Iter: "+str(self.it)+"  Cost: "+str(self.cost))
            i+=1
        return self.arrayTheta
```

A més de tenir tot el codi d'una manera més neta, també hem afegit el concepte de **regulador** que intervé en el càlcul del cost.

Aleshores vam testejar si funcionava com esperavem provant d'executar-lo amb els dos atributs que considerem principals per aquest dataset:





Podem observar com ens dóna un resultat esperat en funció de les dades que tenim. Cal destacar que el contingut és diferent als anteriors plots doncs en aquest hem utilitzat totes les dades disponibles per a mostrar la recta de regressió que genera.

Ha estat en aquest punt quan ens hem disposat a analitzar el comportament del nostre regressor al modificar els paràmetres del descens del gradient:

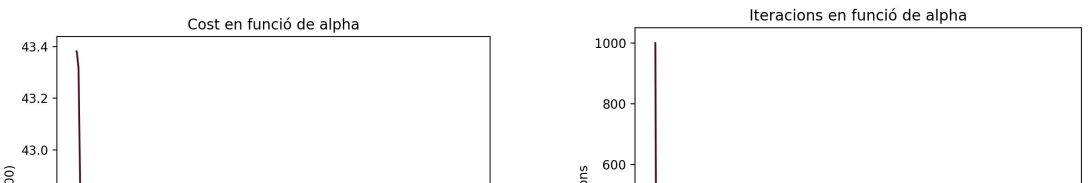
METODOLOGIA EMPRADA

Per als següents estudis hem utilitzat el mateix codi, el que variava era el paràmetre a estudiar.

```
for parametre in testParametre:
    regr = RegressorO(thetas,alpha,X_b,y, it, epsilon, reg)
    #substituim un dels parametres per parametre ^
    thetas = regr.train()
    costosEstudi.append(regr.costos[-1])
    iteracionsPerEstudi.append(regr.it)

#Cost en funció del paràmetre
plt.plot(testParametre,costosEstudi, '#461220' )
plt.title("Cost en funció del parametre")
plt.xlabel("parametre")
plt.ylabel("cost (x100)")
#plt.xscale('log') #opcional
plt.show()

#Iteracions en funció del paràmetre
plt.plot(regTest,iteracionsPerEstudi, '#461220' )
plt.title("Iteracions en funció del parametre")
plt.xlabel("parametre")
plt.ylabel("iteracions")
#plt.xscale('log') #opcional
plt.show()
```

Primer s'entrena cada regresor en funció del valor del paràmetre (guardat en una llista feta per nosaltres) i guardem el seu cost i el nombre d'iteracions realitzades. Posteriorment fem dos representacions: una pel cost i l'altre per les iteracions.

ALPHA



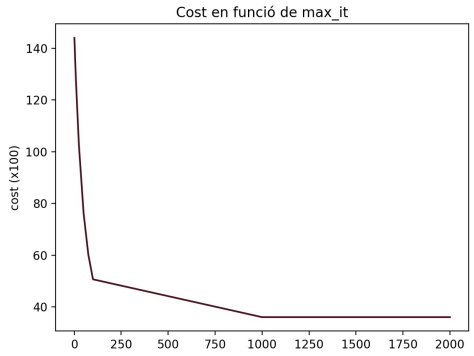
Alpha, com ja s'ha comentat, és el coeficient d'aprenentatge. Amb altres paraules, implica canvis més dràstics o més moderats al modificar les **thetes**. Tot i que el cost no varia dràsticament tenint en compte l'escala dels valors, podem veure **com a menor valor més cost i com a major valor, menys cost**. Això passa perquè inicialment **theta** són valors aleatoris. Si les modifiquem molt poc en un màxim d'iteracions, la seva millora serà insignificativa i tindrà un cos elevat al finalitzar l'entrenament. Pel contrari, amb valors molt elevats els canvis seràn tan dràstics que li costarà trobar un resultat òptim (anirà provant pràcticament valors aleatoris sense arribar a convergir). La primera afirmació la podem veure reflectida en el nostre cas, tot i això la segona no. Hem provat amb alphas encara més gran i el cost acaba sent l'òptim. Si reduissim el nombre d'iteracions màximes, segurament els resultats no serien els mateixos. Per aquest estudi s'ha utilitzat:

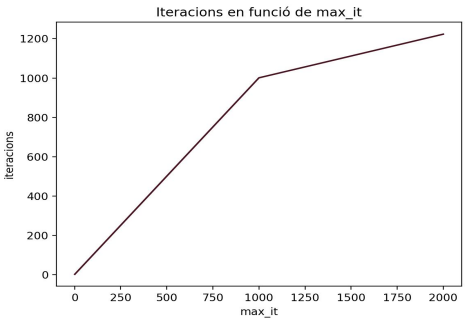
max_iter = 1000 **epsilon** = 1.1e-5 **reg**=10

Les **alphes** a testejar han estat: [0.0001, 0.00025, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1]

També hem volgut analitzar les iteracions realitzades, doncs gairebé sempre convergeix abans d'arribar a la fita. En caquest cas podem veure com per a valors molt petits d'alpha necessita el màxim d'iteracions (i tot i això té un cost elevat respecte les altres alphes). Aquest número d'iteracions es disminueix a mesura que augmentem alpha, però fins a cert punt. Quan superem el valor de 1, el nombre d'iteracions torna a augmentar

ITERACIONS MÀXIMES





Per a aquest segon estudi hem emprat els següents valors pels paràmetres:

alpha = 0.001**epsilon** = 1.1e-5 **reg**=10

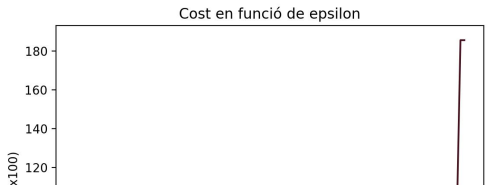
max_it a testejar: [1, 5, 10, 25, 50, 75, 100, 1000, 2000]

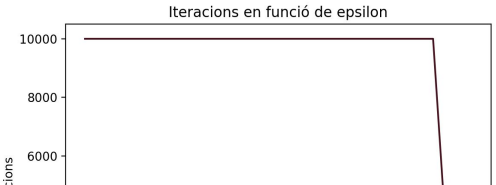
Com es pot observar en l'anterior estudi, a majors alphes les iteracions emprades es redueixen en picat. És per això que hem volgut usar un valor petit per a aquest paràmetre i així obtenir resultats analitzables.

El cost és decreixent en funció del màxim d'iteracions seleccionat. Aquest comportament és evident, doncs segurament si no es poden realitzar les iteracions necessaries per a convergir, el model ens donarà una funció amb més error. Li faltaria poder entrenar més. A més iteracions permeses, menor és el cost i podem intuïr que el model troba una solució òptima, de cost mínim.

Aquesta hipòtesis la podem afirmar amb la segona gràfica. Per a valors petits de max_it, les iteracions realitzades coincideixen amb la fita. Tot i això, per a aquest cas, veiem com amb el màxim a 2000 iteracions el model convergeix abans i s'atura al voltant de les 1200. Podem concloure que la fita òptima per als paràmetres de l'estudi es troba entre 1000 i 2000 iteracions.

EPSILON





Per a aquest segon estudi hem emprat els següents valors pels paràmetres:

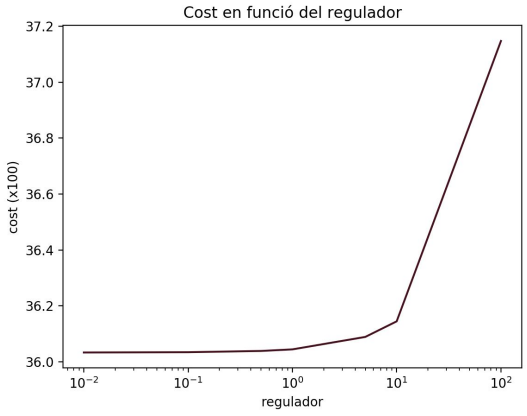
alpha = 0.001**max_it** = 1.1e-5 **reg**=10

epsilon a testejar: 1.0e-100, 1.0e-75, 1.0e-50, 1.0e-25, 1.0e-10, 1.0e-5, 1.0e-2, 1.0e-1, 1.0]

Epsilon és l'error mínim a partir del qual considerem que el nostre model ha convergit en una solució òptima. Contra més petit sigui, més precisa volem que sigui aquesta convergència. És per això que per a valors molt petits d'epsilon tenim costos tant baixos: es busca que la diferència entre un cost i l'anterior sigui molt precisa. En canvi, quan li donem valors prou grans (al voltant de 1, tenint en compte les magnituds del nostre problema) el cost es dispara. Això és degut a que com considerem solució quan la diferència entre dos costos és força gran, el nostre model s'aturarà força aviat.

Aquesta afirmació s'evidencia en el segon gràfic: per a èpsilons petits s'exhaureixen totes les iteracions per a trobar la solucio amb màxima precisió (tenint un cost mínim) i per a èpsilons majors les iteracions es redueixen dràsticament alhora que augmenta en la mateixa magnitud l'error comès.

REGULADOR

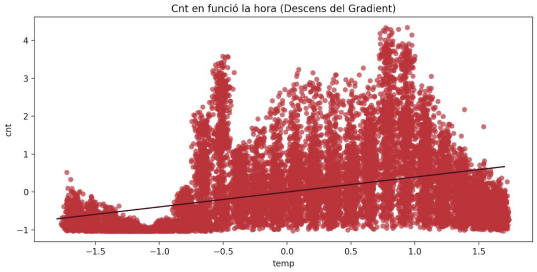


El regulador és el paràmetre que influeix en el cost del regressor en funció del seu valor i el de les thetes. D'aquesta manera ens pot facilitar la cèrca d'ún mínim.

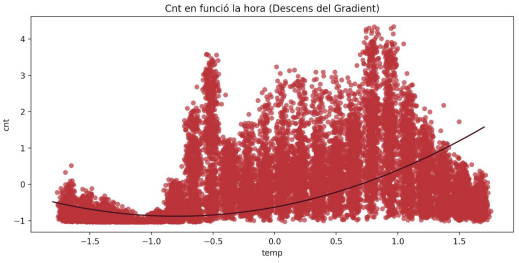
En el nostre cas podem observar com per a valors molt petits el cost és mínim i per a valors més grans, el cost augmenta. Cal destacar, però que aquest augment es força petit tenint en compte l'increment que s'aplica en el regulador.

Per a aquest últim estudi no hem aportat la gràfica de les iteracions en funció del valor del regulador, doncs ens donava un valor constant per a tots els casos. Considerem que no és un resultat a esperar, doncs a menors costos, abans es deu arribar a la convergència si el màxim d'iteracions és constant. Malauradament no hem trobat una explicació per a aquest fenomen.

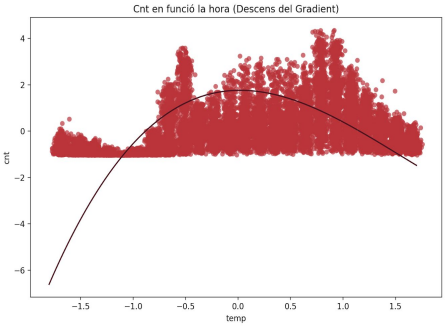
El següen test que s'ha considerat oportú a realitzat és el de testejar funcions no lineals. En el nostre cas hem usat l'atribut hour, ja que hem considerat que una funció polinomial pot fer-ne una regressió més propera a la seva forma que no pas amb temp. Aquests han estat els resultats.

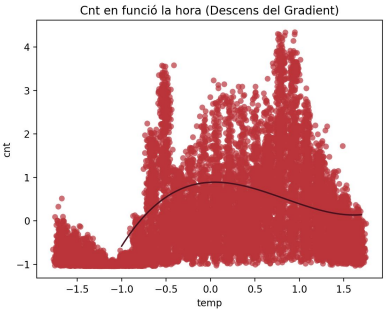


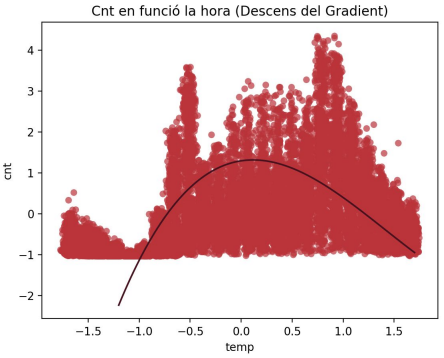
Funció de 1r grau



Funció de 2n Grau







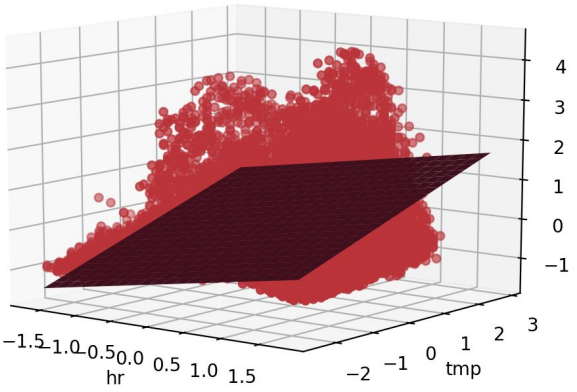
Funció de 4t grau

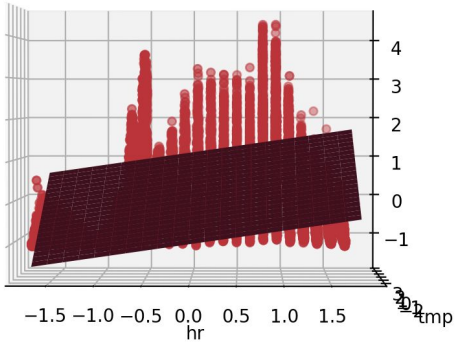
Com podem veure, a mesura que anem augmentant el grau del polinomi, obtenim funcions que representen amb més precisió la distribució de les dades. Sobretot les de 4t grau obtenen un resultat especialment bo. Un detall important

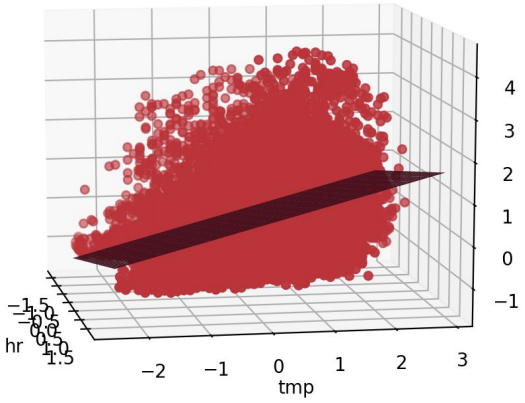
a destacar és que degut a que s'augmenten les dimensions del descens del gradient, l'existència de mínims locals augmenta. És per això que el nostre model, sense variar-ne els paràmetres, ens pot donar funcions amb certes diferències entre un entrenament i un altre. Per aquesta raó hem afegit tres mostres de la funció de 4t grau on tot i que cada una a la vista i a les dades és una funció força representativa de la mostra, entre elles tenen vàries diferències. Probablement, cada una d'elles ha sorgit d'un entrenament que ha arribat a convergència en un mínim local diferent.

Per últim, aprofitant que tenim dos atributs força òptims per a la regressió, ens disposem a fer una regressió en dos dimensions. Per tant el regressor no serà una recta, sino un pla. Per complir amb aquest objectiu usarem el codi per a fer plots 3D del jupyter notebook de l'enunciat.

Com és predecible, els atributs per a entrenar el regressor són **temp** i **hr**. El resultat obtingut és el següent:


4
3
2
1
0
−1
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
hr
−2 −1 0 1 2 3
tmp


4
3
2
1
0
−1
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
hr
tmp


4
3
2
1
0
−1
hr
−2 −1 0 1 2 3
tmp

APARTAT A: CLASSIFICACIÓ

Per la última part del projecte, la tasca és entrenar dos models nous, un **regressor logístic** i una **màquina de vectors de suport**. Aquests dos models són utilitzats per a classificació, pel que l'anterior base de dades no ens és útil pel nostre anàlisi.

És per això que en aquest tercer apartat, canviem el nostre dataset. Aquest a partir d'ara serà el fitxer **mushrooms.csv**, extret del repositori de Machine Learning de la UC Irvine. Aquest és un dataset característic per treballar la classificació, i amb raó, ja que tots els seus atributs són categòrics i balancejats.

A nivell de contingut, la base de dades tracta, com el nom indica, de bolets. Inclou 8124 instàncies corresponents cada una a una mostra d'un bolet diferent, i d'aquests conté 23 atributs diferents, que dónen informació com la olor que fa, com d'abundant és el bolet a la zona, el color del seu vel o si és verinós o no.

Amb un anàlisi ràpid del significat de les variables és fàcil concloure que l'atribut objectiu, i sobre el que s'aplicarà la classificació, és l'últim mencionat, que pot prendre dos valors, **poisonous (p)**, verinós, i **edible (e)**, comestible.

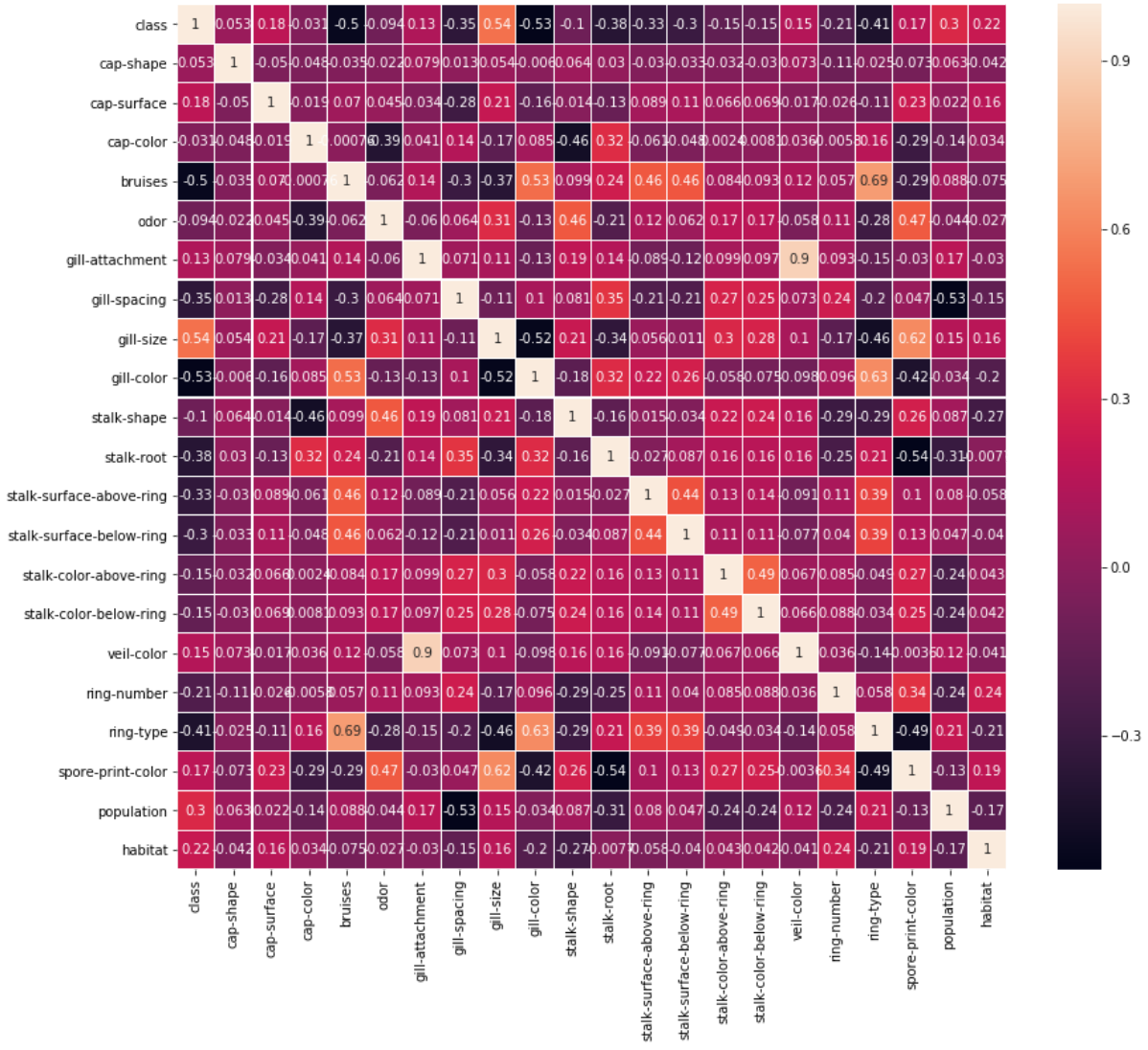
Abans de seguir, comprovem que la base de dades estigui balancejada, ja que si no ho està, no té gaire sentit aplicar classificació, i si es fa s'hauria de tenir molt en compte que un dels possibles valors del target té més pes que l'altre, o bé cambiar l'atribut objectiu, tot i que aquesta última opció hi ha bases de dades que no ho permeten, com és la nostra.

Un cop clar que la nostra base de dades és vàlida per a la classificació, podem seguir amb el següent pas: tractar els valors desconeguts. Els primers anàlisis

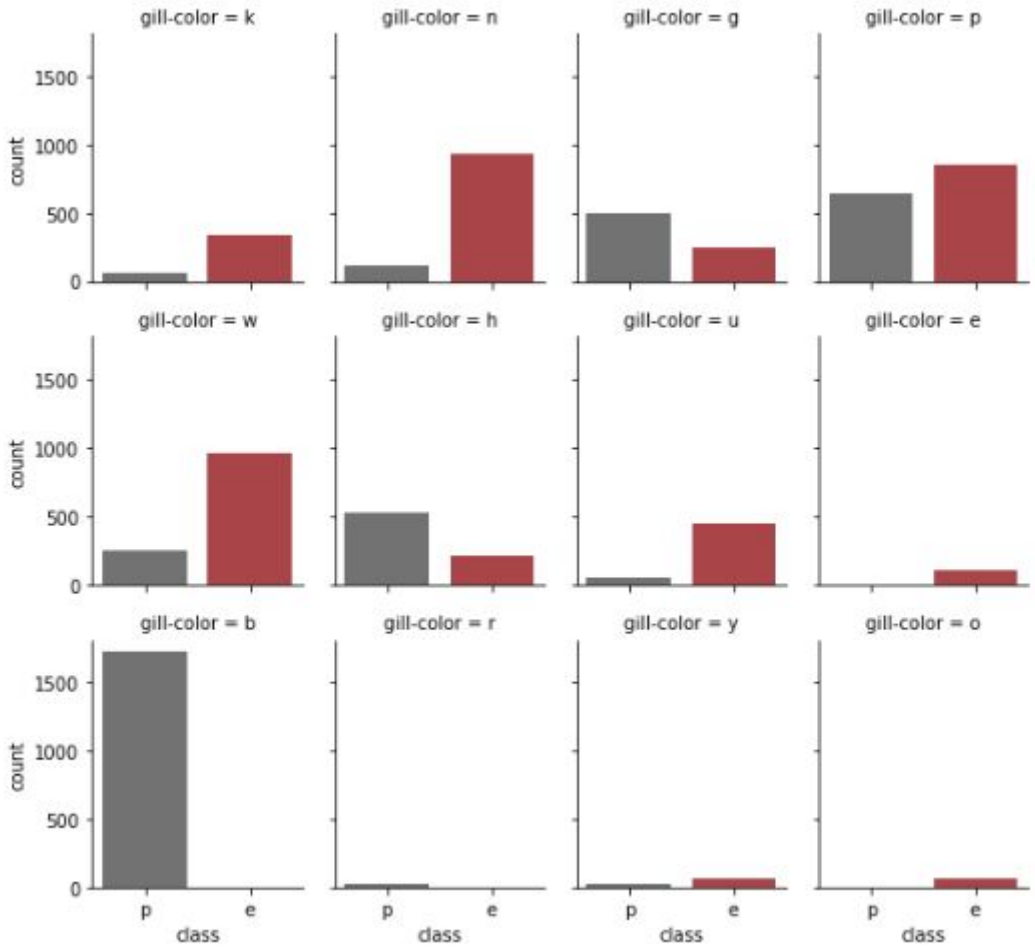
amb les eines de *pandas* semblaven mostrar que no n'hi havia, però mirant millor la documentació de la base de dades vam trobar que era perquè enlloc de marcar-los com a NaN, els marca amb un interrogant i per això no els detectàvem. Un cop solucionat això, vam veure que l'atribut **stalk-root** té pràcticament la meitat dels seus valors desconeguts, i per tant el vam eliminar de la base de dades. També vam veure que l'atribut **veil-type** només contenia valors d'una única classe, pel que és inútil per la classificació, i el vam eliminar també.

Com a últim

Un cop posada a lloc la base de dades, vam analitzar una mica més profundament els atributs, sobretot amb el mapa de calor:



D'aquí, la informació més rellevant és que l'atribut **grill-color** és el que té una major correlació amb la classifiació, amb un pes de -0,53. Si l'analitzem una mica més profundament, veiem perquè passa això:



Com es pot observar a les taules dels diferents valors de la variable, n'hi ha alguns, sent el més notable quan el color és **b**, on és molt clara la classificació, i és per això que veiem tanta correlació al heatmap. Podem concloure que tindrà un pes important aquest atribut a l'hora de fer la classificació.

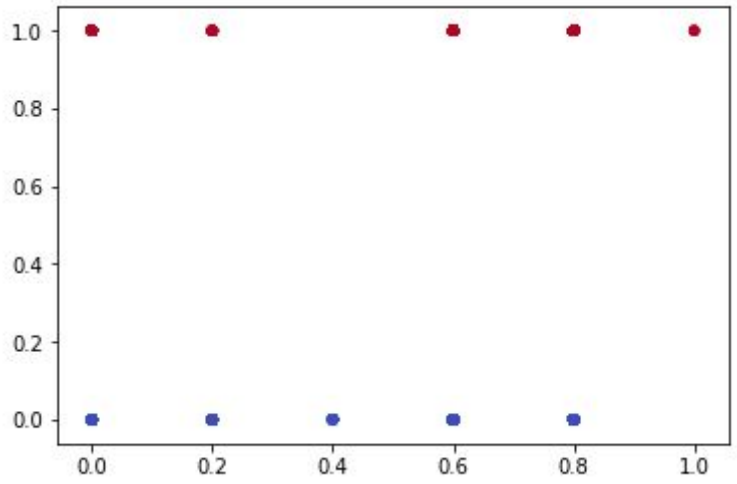
Un cop analitzades de manera empírica les dades, podem transformar-les per facilitar la tasca als algorismes. Això ho fem amb una funció que transforma els valors dels atributs categòrics en continus, de la manera que si un atribut pot agafar 3 valors diferents, se li assignarà al primer un 0, al segon un 0.5 i al tercer un 1. A més, en aquesta conversió també es té en compte que si posem els valors entre 0 i 1, en certa manera ja els estem normalitzant, pel que ja no cal fer aquest pas. La base de dades queda, doncs, de la següent manera:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	...
1	0	0.000	0.000	0.111	0.000	0.125	0.000	0.000	1.000	0.000	...
2	0	0.200	0.000	0.222	0.000	0.250	0.000	0.000	1.000	0.091	...
3	1	0.000	0.333	0.222	0.000	0.000	0.000	0.000	0.000	0.091	...
4	0	0.000	0.000	0.333	1.000	0.375	0.000	1.000	1.000	0.000	...
8119	0	0.800	0.000	0.000	1.000	0.375	1.000	0.000	1.000	0.909	...
8120	0	0.000	0.000	0.000	1.000	0.375	1.000	0.000	1.000	0.909	...
8121	0	0.600	0.000	0.000	1.000	0.375	1.000	0.000	1.000	0.091	...
8122	1	0.800	0.333	0.000	1.000	0.750	0.000	0.000	0.000	0.727	...
8123	0	0.000	0.000	0.000	1.000	0.375	1.000	0.000	1.000	0.909	...

On podem veure alguns dels diversos valors que poden prendre aquests atributs. Per exemple, veiem com **bruises** pot prendre dos valors, **true** o **false**, i es convertexien en 0 i 1. I per altra banda, **gill-color**, que ja hem vist que té un domini de 12 possibles valors, es converteixen en 0, 0.091, 0.727, etc.

Ara sí, era el moment d'aplicar els algorismes.

Primer, mostrem els valors de l'atribut objectiu, on es diferencien clarament les dues classes:



3. CONCLUSIONS

Primerament, fem constància de la gran importància que té fer un estudi profund del dataset, per entendre bé el comportament de les dades i la relació que hi ha entre elles. D'aquesta manera és més fàcil també la comprensió dels resultats obtinguts, ja que es poden comprovar les hipòtesis realitzades a partir del coneixement de les dades. Aquestes també són un aspecte molt important i necessari, com a estudi previ a qualsevol tipus de predicció, perquè ajuden a triar quin tipus d'estudi es vol aplicar sobre la mostra. Hi ha moltíssimes combinacions a escollir, i és molt més fàcil si ja tens una idea prèvia de com poden anar els resultats.

Pel que fa a la regressió lineal, cal destacar la rellevància de la tria d'atributs, que poden afegir soroll o massa precisió al model del regressor. Hem après que s'han de prioritzar aquells atributs que segueixin distribucions normals, i per conseqüència, sempre seran millors per a la regressió aquells que tinguin valors continus i no pas categòrics. A partir d'un anàlisis més empíric, també es poden intuir atributs que tindran més pes, com per exemple que si fa calor, augmentarà el consum d'aigua.

L'anterior entrega ens vam quedar en un punt intermig al resoldre l'apartat B. En aquesta darrera ocasió ja hem pogut aplicar tots els conceptes referents a aquest punt de la pràctica de manera satisfactòria. És cert que hem hagut de reestructurar una mica el codi que utilitzavem, però ha sigut bàsicament transformar la funció creada previament en un objecte amb varis mètodes. També s'han presentat diferents obstacles nous en l'implementació. Un d'ells ha estat a l'hora d'analitzar el comportament per a diferents valors d'un paràmetre, doncs semblava que sempre millorava el model a valors més elevats. Això es deu al fet de que reutilitzavem el valor de thetes generat pel model anterior i per tant començavem l'entrenament en un mínim. Però tal i com es pot observar en aquesta memòria, aquest error fou solucionat i va ser a l'hora d'entendre el perquè del comportament. Els resultats no coincidien gens amb la hipòtesis i per tant es va revisar fins a trobar un possible error. També s'ha acabat de complementar el descens del gradient utilitzant funcions polinomials de grau major a 1 i representant un model que utilitzés dos atributs i per tant la funció fos un pla.

Per tant d'aquest segon apartat, les idees més importants a destacar són la importància de la bona definició dels paràmetres del descens per a un dataset determinat, doncs poden variar el resultat dràsticament. Per altra banda la important possibilitat de no utilitzar un model lineal, doncs en moltes situacions una recta probablement generi uns resultats força pobres al aplicar regressió.

En quant a la classificació, ens quedem amb un mal gust a la boca per no haver pogut assolir els objectius que ens havíem marcat, ja que com es pot veure a la memòria com que hem passat a una base de dades nova ha estat com si

tornéssim a començar, i ens ha ocupat bona part del temps dedicat a aquest apartat entendre-la per poder-la tractar. Sumant això al poc temps que hem pogut dedicar, i a que no l'hem gestionat del tot correctament, aquest apartat es queda coix, comparat amb els altres dos. Estem orgullosos de la feina assolida però creiem que haguéssim pogut arribar molt més enllà, ja que no hem pogut analitzar els resultats de l'entrenament, que és l'interessant d'aquesta assignatura, i d'aquest projecte.

Com a conclusions generals, estem orgullosos del treball realitzat, i que hem après a utilitzar eines que fins ara ens eren totalment desconegudes i a aplicar-les a usos reals, com els que ens podem trobar algun dia al món laboral. A mode d'autocrítica, però, també cal mencionar la gestió del temps, que potser no ha estat l'adequada. Tot i això, creiem que la feina feta és de bona qualitat i com ja hem comentat, trobem que les solucions aportades han estat satisfactòries.